Evolving Long Short-Term Memory Networks*

Vicente Coelho Lobo Neto[0000−0001−8593−9583], Leandro Aparecido Passos[0000−0003−3529−3109], and João Paulo Papa[0000−0002−6494−7514]

Recogna Laboratory, School of Sciences, São Paulo State University, Bauru, Brazil
{vicente.lobo, leandro.passos, joao.papa}@unesp.br
www.recogna.tech

Abstract. Machine learning techniques have been massively employed in the last years over a wide variety of applications, especially those based on deep learning, which obtained state-of-the-art results in several research fields. Despite the success, such techniques still suffer from some shortcomings, such as the sensitivity to their hyperparameters, whose proper selection is context-dependent, i.e., the model may perform better over each dataset when using a specific set of hyperparameters. Therefore, we propose an approach based on evolutionary optimization techniques for fine-tuning Long Short-Term Memory networks. Experiments were conducted over three public word-processing datasets for part-of-speech tagging. The results showed the robustness of the proposed approach for the aforementioned task.

Keywords: Long Short-Term Memory · Part-of-Speech tagging · Metaheuristic Optimization · Evolutionary Algorithms.

1 Introduction

Machine learning techniques achieved outstanding outcomes in the last years, mostly due to the notable results obtained using deep learning techniques in a wide variety of applications [1], ranging from medicine [2] to route obstruction detection [3]. In this context, a specific kind of model, the so-called Recurrent Neural Network (RNN), has been extensively employed to model temporal-dependent data, such as stock market forecasting and applications that involve text and speech processing.

Such networks are time-dependent models whose neurons receive recurrent feedback from others, in contrast to densely connected models (e.g., Multilayer Perceptron). In short, the model's current output at time t depends on its input as well as the output obtained at time $t-1$. Due to this intrinsic characteristic, this kind of network deals very well with data that has temporally-related information, such as videos, audio, and texts.

* The authors are grateful to FAPESP grants #2013/07375-0, #2014/12236-1, #2017/25908-6, #2018/10100-6, #2019/07665-4, as well as CNPq grants #307066/2017-7 and #427968/2018-6.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

Among a wide variety of RNNs, such as the Hopfield Neural Network [4] and the Gated Recurrent Unit Networks [5], the Long Short-Term Memory network, also known as LSTM [6] has drawn considerable attention in the last years. LSTM is an RNN architecture designed to model temporal sequences and their long-range dependencies more accurately than conventional RNNs. It obtained impressive results over a variety of time-series problems, such as named entity recognition [7], chunking [8], acoustic modelling [9], and Part-of-Speech (PoS) tagging [7], to cite a few. However, the method still suffers from a well-known problem related to neural network-based approaches: the proper selection of their hyperparameters. A few years ago, Greff et al. [10] used a random search to handle such a problem. Later on, Reimers e Gurevych [11] studied how these hyperparameters affect the performance of LSTM networks, demonstrating that their optimization is crucial for satisfactory outcomes.

Recently, many works addressed the problem of hyperparameter fine-tuning in neural networks using nature-inspired metaheuristic optimization techniques. Fedorovici et al. [12], for instance, employed the Gravitational Search Algorithm [13] to optimize Convolutional Neural Networks (CNN) [14], while Rosa et al. [15] proposed a similar approach using Harmonic Search [16]. Later, Rosa et al. [17] used the Firefly algorithm [18] to fine-tune Deep Belief Networks [19], as well as Passos et al. [20, 21] applied a similar approach in the context of Deep Boltzmann Machines [22].

However, only a very few works employed evolutionary optimization techniques to fine-tune LSTM networks to date [23, 24], but none concerning the PoS tagging task. Therefore, the main contribution of this paper is to fill up this gap by introducing evolutionary techniques for LSTM hyperparameter fine-tuning in the context of Part-of-Speech tagging. For such a task, we compared four evolutionary-based optimization approaches: Genetic Algorithm (GA) [25], Genetic Programming (GP) [26], Geometric Semantic Genetic Programming (GSGP) [27], and the Stack-based Genetic Programming (SGP) [28], as well as a random search as the baseline.

The remainder of this paper is organized as follows. Section 2 briefly introduces the main concepts of LSTM and PoS tagging, while Section 3 describes the approach considered for LSTM hyperparameter fine-tuning. Further, the methodology is detailed in Section 4, while the results are presented in Section 5. Finally, Section 6 states conclusions and future works.

2 Theoretical Background

This section briefly introduces the main concepts regarding the Part-of-Speech tagging, as well as the Long Short-Term Memory Networks.

2.1 Part-of-Speech Tagging

Generally speaking, a sentence is a composition of words joined according to some rules, whose intention is to express a complete idea. In such a context,

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

each word in a sentence performs a specific function, e.g., a verb indicates an action, an adjective adds some attribute to a noun, and the latter is used to name any being.

Natural languages, such as English, Portuguese, and any other, possess a considerable amount of words. Therefore, it is essential to note that many of them may admit ambiguous meanings, i.e., the same word can hold two or more different connotations. To illustrate the idea, suppose the word "seed": it may stand for a noun, referring to the reproductive unity of a plant, as well as denote a verb, representing the act of sowing the soil with seeds.

To handle such ambiguous meanings, one can make use of the Part-of-Speech tagging process. The method aims at, given a sentence, identifying each compounding word's grammatical function inside it, as depicted in Figure 1.

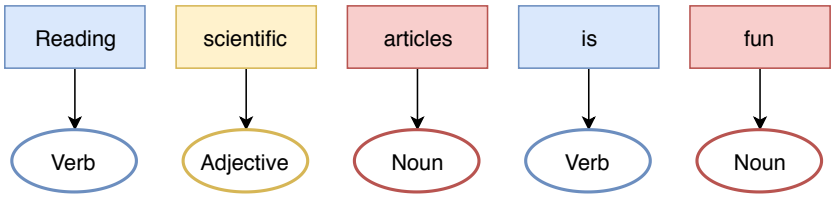

Fig. 1: Example of PoS tagging.

Understanding the true meaning of a word in a sentence is essential to comprehend the message conveyed through it thoroughly. Though it stands for a trivial task for human beings, automatic PoS tagging presents itself as quite challenging for computer systems. Among the various ways of solving this task, one can cite rule-based algorithms [29], as well as Hidden Markov Models [30].

Finally, one should notice the importance of contextual information in order to assign a tag to a word correctly. One approach that yields good results while addressing such context employs machine learning techniques through Recurrent Neural Networks. Therefore, the next section introduces the LSTM, the current state-of-art approach for natural language processing.

2.2 Long Short-Term Memory Network

Although LSTMs present an architecture similar to traditional RNNs methods, they employ a different function for computing their hidden states. Such modification is essential for avoiding problems such as the well-known vanishing gradient, which can occur while relating current information to long-term network knowledge, known as long-term dependency. LSTM addresses this problem by using structures called **cells**, which receive as input their previous states as well as the current input. Internally, these cells decide what should be kept and forwarded to the next iterations, and what should be erased from memory.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

Therefore, they are capable of combining input information with previous states, as well as the current memory.

A cell is composed of states, as well as three gates: (i) the *input gate*, which is responsible for deciding what information is relevant and how much of it will be updated or deleted; (ii) the *forget gate*, used when the network needs to erase any previous information present in the cell's memory in order to store a new document; and (iii) the *output gate*, that is employed to compute the cell's state [31]. Besides, every gate is activated by a sigmoid function, and the cell's output is subject to a combination of the gates' outputs, which will decide whether information should be forgotten or kept in the cell's memory.

Let $\boldsymbol{f}^t,\ \boldsymbol{i}^t,\ \boldsymbol{o}^t \in \Re^{m\times 1}$ be the outputs of the forget, input, and output gates at time step t, respectively, which can be computed as follows:

$$\boldsymbol{f}^t = \sigma(\boldsymbol{W}_f^t \boldsymbol{x}^t + \boldsymbol{U}_f^t \boldsymbol{h}^{t-1} + \boldsymbol{b}_f^t), \tag{1}$$

$$\boldsymbol{i}^t = \sigma(\boldsymbol{W}_i^t \boldsymbol{x}^t + \boldsymbol{U}_i^t \boldsymbol{h}^{t-1} + \boldsymbol{b}_i^t), \tag{2}$$

and

$$\boldsymbol{o}^t = \sigma(\boldsymbol{W}_o^t \boldsymbol{x}^t + \boldsymbol{U}_o^t \boldsymbol{h}^{t-1} + \boldsymbol{b}_o^t), \tag{3}$$

where $\boldsymbol{x}^t \in \mathbb{R}^{n\times 1}$ denotes the input vector at time step t, $\boldsymbol{h}^{t-1} \in \mathbb{R}^{m\times 1}$ stands for the output of the previous cell, and $\boldsymbol{W}_f^t, \boldsymbol{W}_i^t,\ \boldsymbol{W}_o^t \in \mathbb{R}^{m\times n}$ and $\boldsymbol{U}_f^t, \boldsymbol{U}_i^t, \boldsymbol{U}_o^t \in \mathbb{R}^{m\times m}$ denote the weight matrices at time step t. Finally, $\boldsymbol{b}_f^t, \boldsymbol{b}_i^t, \boldsymbol{b}_o^t \in \mathbb{R}^{m\times 1}$ correspond to the biases of each gate at time step t.

Further, one can compute the cell's state $\boldsymbol{c}^t \in \mathbb{R}^{m\times 1}$ at time step t considering the forget and input gates' output as follows:

$$\boldsymbol{c}^t = \boldsymbol{f}^t \otimes \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \otimes tanh(\boldsymbol{W}_c^t \boldsymbol{x}^t + \boldsymbol{U}_c^t \boldsymbol{h}^{t-1} + \boldsymbol{b}_c^t), \tag{4}$$

where $\otimes$ stands for the Hadamard product, and $\boldsymbol{W}_c \in \mathbb{R}^{m\times n}$ and $\boldsymbol{U}_c \in \mathbb{R}^{m\times m}$ are the weight matrices, and $\boldsymbol{b}_c \in \mathbb{R}^{m\times 1}$ corresponds to the bias of the cell's state. Therefore, one can compute the output of a cell at time step t as follows:

$$\boldsymbol{h}^t = \boldsymbol{o}^t \otimes tanh(\boldsymbol{c}^t). \tag{5}$$

Finally, the weights of an LSTM network are updated using a gradient-based backward pass as follows:

$${}^z\boldsymbol{W}_f^{t+1} = {}^z\boldsymbol{W}_f^t - \alpha({}^z\delta_f^t \otimes \boldsymbol{h}^t), \tag{6}$$

$${}^z\boldsymbol{W}_i^{t+1} = {}^z\boldsymbol{W}_i^t - \alpha({}^z\delta_i^t \otimes \boldsymbol{h}^t), \tag{7}$$

and

$${}^z\boldsymbol{W}_o^{t+1} = {}^z\boldsymbol{W}_o^t - \alpha({}^z\delta_o^t \otimes \boldsymbol{h}^t), \tag{8}$$

where $\alpha \in \mathbb{R}$ is the learning rate and ${}^z\boldsymbol{W}_f^t$, ${}^z\boldsymbol{W}_i^t$, and ${}^z\boldsymbol{W}_o^t$ stand for the z^{th} column of matrices $\boldsymbol{W}_f^t$, $\boldsymbol{W}_i^t$, and $\boldsymbol{W}_o^t$, respectively. Notice that $\delta_f^t, \delta_i^t, \delta_o^t \in \mathbb{R}^{m\times n}$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

denote the partial derivatives at time step t, and ${}^{z}\delta_f^t$, ${}^{z}\delta_i^t$, and ${}^{z}\delta_o^t$ correspond to the z^{th} column of matrices ${}^{z}\delta_f^t$, ${}^{z}\delta_i^t$, and ${}^{z}\delta_o^t$, respectively. Further, the bias of each gate is also updated in a similar fashion, and the gradient is propagated to the former layers using the backpropagation chain's rule. Finally, the weight matrices $\boldsymbol{U}_f^t$, $\boldsymbol{U}_i^t$, and $\boldsymbol{U}_o^t$ are updated similarly to their respective counterparts, i.e., Equations 6, 7, and 8.

3 LSTM fine-tuning as an optimization problem

In the context of parameterized machine learning techniques, we must refer to two different denominations: (i) *parameters* and (ii) *hyperparameters*. Typically, the first term represents low-level parameters that are not controlled by the user, such as connection weights in neural networks, for example. The other term refers to high-level parameters that can be adjusted and chosen by the user, such as the learning rate and the number of layers, among others. Both terms are of crucial importance for improving the performance of neural models.

Regarding LSTM parameter optimization, the well known back-propagated using gradient descent present itself as a suitable method for the task. However, a proper selection of its hyperparameters poses a more challenging task since it requires the user a previous knowledge of the problem. Hence, an automatic method to select the model most relevant hyperparameters is strongly desirable, i.e., the *learning rate*, which denotes the step-size towards the gradient direction, the *embedding layer size*, which is responsible for the representation of words and their relative meanings, and the *LSTM size* itself, which denotes the number of LSTM's cells composing the model.

Therefore, this work proposes employing evolutionary algorithms to fine-tune such hyperparameters. These methods are capable of automatically selecting a good set of hyperparameters by randomly initializing a set of candidate solutions that evolve over time. In this context, the candidate solutions are represented by a vector, where each position denotes a hyperparameter to be fine-tuned. In this work, for instance, it is considered a 3-dimensional representation, denoting the learning rate, the embedding layer size, and the LSTM size.

Further, every solution is evaluated after each evolution cycle. This evaluation consists of training the LSTM network using both the candidate solution set of hyperparameters together with the training set. Afterward, the trained model is employed to classify the evaluation set, and the loss function obtained over this procedure, denoted fitness function, is used to evaluate and update the solutions towards a minimum (ideally, the global minimum). At the end of the process, it is expected that the model learns a reasonably good set of hyperparameters. Figure 2 depicts the model adopted in the work.

3.1 Evolutionary Optimization Algorithms

This section briefly describes the four metaheuristic optimization techniques employed in this work concerning the task of LSTM hyperparameter fine-tuning.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

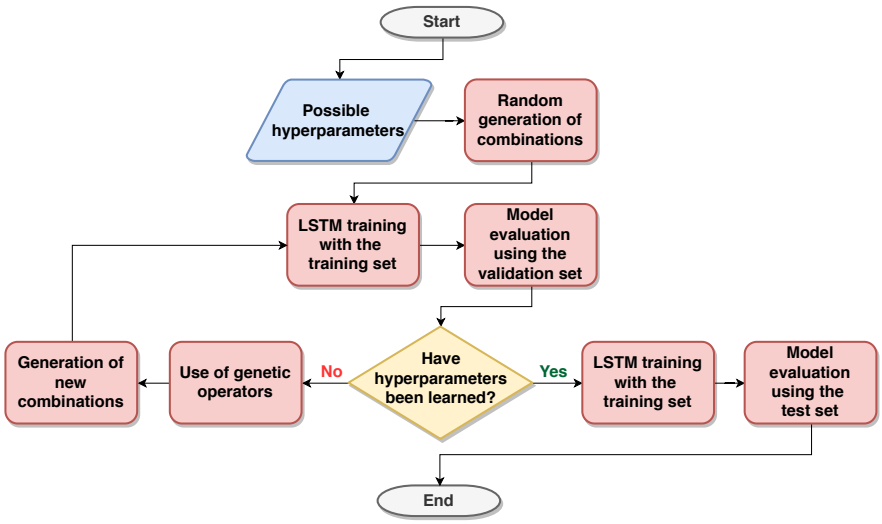

Fig. 2: Pipeline employed in the work.

3.1.1 Genetic Algorithm: GA is a specific case of evolutionary algorithms that employ concepts and expressions from natural genetics that models each possible solution as a chromosome. Among a variety of implementations of the model, this work represents each individual as an array composed of the hyperparameter to be optimized, letting aside the binary version. Moreover, the following evolution operators were employed:

1. **Elitism**: it retains the individuals with the best fitness values of the current population using elitism to accelerate the convergence;
2. **Random selection**: it adds some random elements of the current population to the new one to promote greater genetic diversity;
3. **Mutation**: it modifies a random position of individuals of the new population generating values within the allowed limits; and
4. **Crossover**: it randomly selects two individuals from the population and creates two new individuals through the single-point crossover.

3.1.2 Genetic Programming: GP employs the very same idea behind the GA, except for data structure, which is represented by an expression tree. In this model, each tree's leaf node contains an array, similar to the ones applied in GA, representing the hyperparameters to be optimized. Further, inner-nodes stand for mathematical operators, such as addition, multiplication, and logarithm, among others.

3.1.3 Geometric Semantic Genetic Programming: Despite the advantages of GP, since it uses purely syntactic operators, instead of the values of

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

the individuals' decision variables while applying the operators, it may perform some very abrupt changes in the values of the variables as the evolution process occurs, which may adversely affect the convergence.

Therefore, GSGP considers the evolution of individuals by selecting semantically close solutions [27]. The procedure is performed by introducing some restrictions in the mutation and crossover operators.

3.1.4 Stack-based Genetic Programming: The SGP is a variation of GP that, instead of using a tree data structure, it employs two stacks: one for data input and one for the output. Thus, the SGP input stack is composed of a set of operations and operands responsible for describing a mathematical expression, similar to GP, while the output stack contains the expression evaluation result [28].

Since there are no syntactic constraints in this technique, individuals can be represented as vectors. Besides, it is possible to use the same genetic operators as the standard Genetic Algorithm for mutation and crossover. Thence, the technique combines the great individuals' variability of GP with the simplicity of GA's genetic operators, thus providing better results than GP, in general, depending on the implementation and the problem addressed.

4 Methodology

This section presents the methodology employed in the experiments, i.e., the datasets, experimental setup, and network modeling concerning the PoS tagging problem.

4.1 Datasets

This work employs three well-known public datasets for the task of Part-Of-Speech tagging. Such datasets are composed of natural language text fragments, and they are available at the Natural Language Toolkit (NLTK) library [32]:

- *Brown Corpus of Standard American English* (**Brown**): approximately one million English text sentences divided into 15 different categories, including newspapers, scientific articles, and other sources.
- *Floresta Sintá(c)tica* (**Floresta**): sentences in Brazilian Portuguese extracted from interviews' transcriptions, parliamentary discussions, literary texts, and articles from the Brazilian newspaper Folha de São Paulo compose this corpus.
- *Penn Treebank* (**Treebank**): covering a variety of subjects, the sentences of this corpus are in English and were extracted from three years of The Wall Street Journal.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

4.2 Modeling the PoS tagging problem with LSTM

Since LSTM networks require a vector composed of real numbers as input, the datasets were pre-processed such that the words were encoded into a numerical dictionary, where each word corresponds to a non-zero number. Besides, the input is composed of an n-dimensional vector whose features should fit in, i.e., all sentences should have the same size. Therefore, n was selected based on the longest sentence's size. Finally, a padding approach was employed to fulfill all the non-occupied spaces with zeros.

Further, each sentence is represented by an array of tuples using the pattern (word, PoS tag). Later, these tuples are split, and the tags are converted into their categorical (one-hot) representation to match the LSTM output format.

Therefore, the sequential architecture of the LSTM employed in this work for the task of PoS tagging is composed of the following layers:

1. **Input**: Receives input data. It outputs data with (n, d) shape, where d and n stand for the number of inputs and the number of input features, respectively;
2. **Embedding**: It provides a dense representation of words and their relative meanings, converting the indices into dense vectors. Output data has (n, d, E) shape, where E stands for the embedding layer size;
3. **Bidirectional LSTM layer**: is the core of the network, comprising the LSTM units. The bidirectional version was selected instead of the unidirectional due to the superior results obtained by the model. Output shape is (n, d, $2L$), where L represents the number of LSTM units. As the bidirectional version was used, $2L$ units must be employed;
4. **Time-distributed Dense**: A standard densely connected layer with a wrapper to handle the input data temporal distribution. Considering that the tags were converted to the categorical representation, the output shape of this layer is (n, d, j), with j standing for the number of available tags.

Figure 3 depicts the model mentioned above.

Fig. 3: Model representation with output shapes.

Further, the Softmax is then employed as the activation function. Concerning the loss function, the model aims to minimize the categorical cross-entropy, which is the most commonly applied approach for text classification tasks.

4.3 Experimental Setup

This work compares the performance of four well-known evolutionary optimization algorithms, i.e., GA, GP, GSGP, and SGP, as well as a random search, to

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

the task of LSTM network hyperparameter fine-tuning in the context of PoS tagging. The experiments were performed over 400 randomly selected sentences from three public datasets commonly employed for the task: Brown, Floresta, and Treebank. Further, these sentences were partitioned into three sets: training, validation, and testing using the percentages of 60%, 20% and 20%, respectively.

This work focuses on finding the set of hyperparameters that minimizes the categorical cross-entropy loss function, thus maximizing the LSTM performance for the task of PoS tagging using metaheuristic evolutive algorithms. Therefore, the problem was designed to search for three hyperparameters: (i) the layer size L and (ii) the embedding layer size E, both selected from a closed set of powers of 2 ranging from 2^1 to 2^{10}, i.e., {2, 4, 8, 16, 32, 64, 128, 256, 512, 1024}, and (iii) the learning rate $\alpha \in [0.001, 0.1]$. Since the metaheuristic techniques work with real numbers only, one should convert them to the nearest power of 2 when dealing with L and E.

Concerning the optimization procedure, each technique employed 10 individuals evolving during 20 iterations and considered a retention rate of 20%, a random selection rate of 5%, and 1% of mutation rate. Moreover, GP and GSGP employed expression trees with a maximum tree height of 5, while the SGP employed arrays with 10 elements. Also, they implemented the addition, subtraction, multiplication, division, modulo, negation, logarithm, and square root operators. Notice the latter two were not employed in the GP since they adversely affected convergence[1]. Finally, these techniques were compared against a random search, which stands for the experiments' baseline.

Afterward, the best hyperparameters obtained during the optimization process are employed to train the network for 200 iterations for further classification. Besides, each experiment was performed during 20 runs for statistical analysis through the Wilcoxon signed-rank test [33] with a significance of 5%. Finally, the experiments were conducted using both the Keras open-source library [34] and Tensorflow [35].

5 Experimental Results

This section presents the experimental results. Table 1 provides the mean accuracy and the standard deviation concerning the Brown Corpus of Standard American English, Floresta Sintá(c)tica, and the Penn Treebank datasets. Notice that the best results, according to the Wilcoxon signed-rank test, are presented in bold.

From these results, it is possible to extract two main conclusions: (i) metaheuristic optimization techniques performed better than a random search concerning the three datasets, which validates the employment of such methods for the task; (ii) the tree-based algorithms, i.e., GP, GSGP, and SGP, performed better than GA in two out of three datasets. Such behavior is expected since these techniques employ more robust approaches to solve the optimization problem.

[1] Notice that all these parameters and configurations were set up empirically

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

Table 1: Mean accuracy results over Brown, Floresta and Treebank datasets.

	GA	GP	GSGP	SGP	Random
Brown	**0.734 ± 0.043**	**0.740 ± 0.092**	**0.726 ± 0.100**	**0.673 ± 0.115**	0.566 ± 0.229
Floresta	0.656 ± 0.121	**0.706 ± 0.025**	0.676 ± 0.038	**0.603 ± 0.207**	0.470 ± 0.271
Treebank	0.748 ± 0.052	**0.685 ± 0.202**	**0.782 ± 0.024**	**0.751 ± 0.098**	0.658 ± 0.206

5.1 Training evaluation

Considering the evaluation of the learned models, Figure 4 depicts the mean loss obtained by each model considering the three test sets. One can observe that minimum values do not necessarily reflect in the best accuracy, as depicted in Brown, where GA obtained the minimum loss, and GP obtained the higher accuracy; and the Treebank dataset, where SGP obtained the minimum loss and GSGP obtained the higher accuracies. Nevertheless, all techniques still present mean loss values considerably lower than the random search.

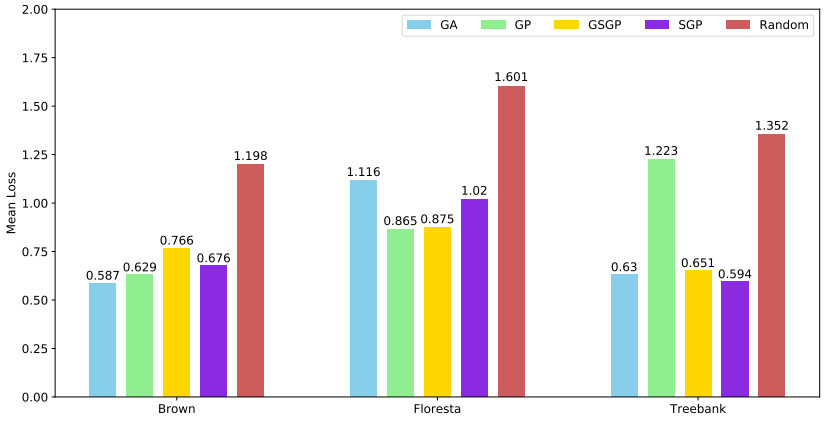

Fig. 4: Mean loss per data set using evolutionary optimization and random selection for LSTM training purposes.

5.2 Optimization Progress

Figure 5 depicts each model's convergence, considering the loss over the validation set, during the training procedure. The figure clearly shows that GA obtained the best values regarding the first iterations considering both Brown and Floresta datasets, depicted in Figures 5(a) and (b), respectively. Additionally, concerning the Treebank dataset, although GSGP obtained the best values

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

in the initial iterations, it is outperformed by GA in the third iteration, whose convergence keeps improving until reaching the stop criteria of 20 iterations. Moreover, it can be observed that GSGP and SGP do not present significant progress during the iterations, getting stuck on local optima and even get worst results than the ones obtained in previous iterations. Finally, considering the GP algorithm, despite the slightly positive convergence, it was not capable of outperforming GA in any of the three datasets.

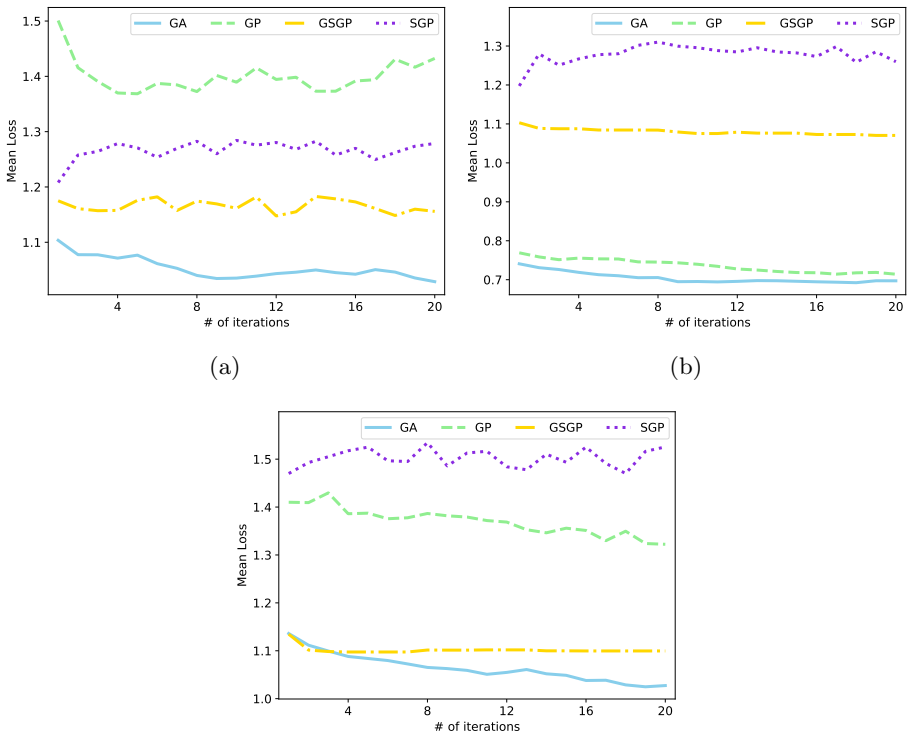

Fig. 5: Validation set loss convergence over the training steps considering: (a) Brown, (b) Floresta, and (c) Treebank datasets.

It is interesting to notice that, although GA outperformed the tree-based methods concerning the validation set over a training composed of 20 iterations when considering the same hyperparameters over the testing set considering 200 iterations for training, the tree-based methods obtained the best results. Such a phenomenon may suggest the model becomes overfitted when trained considering the GA set of hyperparameters.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

6 Conclusions and Future Works

In this paper, we addressed the problem of LSTM hyperparameter fine-tuning through evolutionary metaheuristic optimization algorithms in the context of PoS tagging. For this task, four techniques were compared, i.e., the Genetic Algorithm, the Genetic Programming, the Geometric Semantic Genetic Programming, and the Stack-based Genetic Programming, as well as a random search, employed as the baseline. Experiments conducted over three well-known public datasets confirmed the effectiveness of the proposed approach since all four metaheuristic algorithms outperformed the random search. Further, although GA presented a smoother convergence during the optimization steps, the tree-based evolutionary techniques obtained the best loss and accuracy results when considering a more extended training period over the testing sets. Such results highlight that those complex mathematical operations, like the ones performed by GP, GSGP, and SGP, contribute to the convergence of the model in this context.

Regarding future works, we intend to validate the proposed approach concerning other contexts than PoS tagging, whether textual or not, as well as using different artificial neural networks.

References

1. Y. LeCun, Y.Bengio, G. E. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
2. L. A. Passos, C. Santos, C. R. Pereira, L. C. S. Afonso, J. P. Papa, A hybrid approach for breast mass categorization, in: ECCOMAS Thematic Conference on Computational Vision and Medical Image Processing, Springer, 2019, pp. 159–168.
3. M. C. Santana, L. A. Passos, T. Moreira, D. Colombo, V. H. C. De Albuquerque, J. P. Papa, A novel siamese-based approach for scene change detection with applications to obstructed routes in hazardous environments, IEEE Intelligent Systems.
4. J. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proceedings of the national academy of sciences 79 (8) (1982) 2554–2558.
5. L. C. Ribeiro, L. C. Afonso, J. P. Papa, Bag of samplings for computer-assisted parkinson's disease diagnosis based on recurrent neural networks, Computers in biology and medicine 115 (2019) 103477.
6. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735–1780.
7. X. Ma, E. H. Hovy, End-to-end sequence labeling via bi-directional lstm-cnns-crf, CoRR abs/1603.01354.
8. A. Komninos, S. Manandhar, Dependency based embeddings for sentence classification tasks, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 1490–1500.
9. H. Sak, A. Senior, F. Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in: Fifteenth annual conference of the international speech communication association, 2014, pp. 338–342.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

10. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, J. Schmidhuber, LSTM: A search space odyssey, CoRR abs/1503.04069.
11. N. Reimers, I. Gurevych, Optimal hyperparameters for deep lstm-networks for sequence labeling tasks, CoRR abs/1707.06799.
12. L. Fedorovici, R. Precup, F. Dragan, R. David, C. Purcaru, Embedding gravitational search algorithms in convolutional neural networks for OCR applications, in: 7th IEEE International Symposium on Applied Computational Intelligence and Informatics, SACI '12, 2012, pp. 125–130. doi:10.1109/SACI.2012.6249989.
13. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Information Sciences 179 (13) (2009) 2232–2248.
14. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.
15. G. H. Rosa, J. P. Papa, A. N. Marana, W. Scheirer, D. D. Cox, Fine-tuning convolutional neural networks using harmony search, in: A. Pardo, J. Kittler (Eds.), Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications, Vol. 9423 of Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 683–690, 20th Iberoamerican Congress on Pattern Recognition.
16. Z. W. Geem, Music-Inspired Harmony Search Algorithm: Theory and Applications, 1st Edition, Springer Publishing Company, Incorporated, 2009.
17. G. H. Rosa, J. P. Papa, K. A. P. Costa, L. A. Passos, C. R. Pereira, X.-S. Yang, Learning parameters in deep belief networks through firefly algorithm, in: Artificial Neural Networks in Pattern Recognition: 7th IAPR TC3 Workshop, ANNPR, Springer International Publishing, Cham, 2016, pp. 138–149.
18. X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, International Journal Bio-Inspired Computing 2 (2) (2010) 78–84.
19. G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (7) (2006) 1527–1554.
20. L. A. Passos, D. R. Rodrigues, J. P. Papa, Fine tuning deep boltzmann machines through meta-heuristic approaches, in: 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), IEEE, 2018, pp. 000419–000424.
21. L. A. Passos, J. P. Papa, Temperature-based deep boltzmann machines, Neural Processing Letters 48 (1) (2018) 95–107.
22. R. Salakhutdinov, G. E. Hinton, An efficient learning procedure for deep boltzmann machines, Neural Computation 24 (8) (2012) 1967–2006.
23. H. Chung, K.-S. Shin, Genetic algorithm-optimized long short-term memory network for stock market prediction, Sustainability 10 (10) (2018) 3765.
24. T. Kim, S. Cho, Particle swarm optimization-based cnn-lstm networks for forecasting energy consumption, in: 2019 IEEE Congress on Evolutionary Computation, 2019, pp. 1510–1516.
25. D. E. Goldberg, J. H. Holland, Genetic algorithms and machine learning, Machine Learning 3.
26. J. R. Koza, J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.
27. A. Moraglio, K. Krawiec, C. G. Johnson, Geometric semantic genetic programming, in: PPSN, 2012, pp. 21–31.
28. T. Perkis, Stack-based genetic programming, in: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 1994, pp. 148–153 vol.1. doi:10.1109/ICEC.1994.350025.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25

29. E. Brill, A simple rule-based part of speech tagger, in: Proceedings of the third conference on Applied natural language processing, Association for Computational Linguistics, 1992, pp. 152–155.
30. J. Kupiec, Robust part-of-speech tagging using a hidden markov model, Computer Speech & Language 6 (3) (1992) 225–242.
31. A. Gers F., J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm, in: 9th International Conference on Artificial Neural Networks: ICANN '99, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 1999, pp. 850–855.
32. E. Loper, S. Bird, Nltk: The natural language toolkit, in: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 63–70. doi:10.3115/1118108.1118117.
URL https://doi.org/10.3115/1118108.1118117
33. F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bulletin 1 (6) (1945) 80–83.
34. P. Charles, Project title, https://github.com/charlespwd/project-title (2013).
35. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL http://tensorflow.org/

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50417-5_25